

Elementos de Computação em R para Inferência Estatística

PJ

Ultima atualização: 10 de agosto de 2022

1 Inferência para parâmetros da distribuição Gama

O objetivo deste documento é revisar conceitos de inferência estatística, com ênfase na função de verossimilhança. Tratamos aqui de um caso de modelo com dois parâmetros.

1.1 Densidade

Parametrização 1:

Esta é a parametrização utilizada nas funções `{r,p,q,d}gamma()` do R com:

$\alpha = \text{shape} = a$ e $\beta = \text{scale} = s$ ($\text{rate} = 1/\text{scale}$).

Note que $a = 0$ define uma fdp trivial com toda massa de probabilidade em zero. Existe um alerta importante na documentação da função sobre a capacidade do computador em representar pequenos valores que ocorrem para certos valores dos parâmetros:

Note that for smallish values of shape (and moderate scale) a large parts of the mass of the Gamma distribution is on values of x so near zero that they will be represented as zero in computer arithmetic. So `rgamma` can well return values which will be represented as zero. (This will also happen for very large values of scale since the actual generation is done for `scale=1`.)

Funções e expressões: densidade, esperança, variância, (log)-verossimilhança, escore e hessiano:

$$\begin{aligned} Y &\sim G(\alpha, \beta) \\ f(y|\alpha, \beta) &= \frac{1}{\Gamma(\alpha) \beta^\alpha} y^{\alpha-1} \exp\{-y/\beta\} \quad y \geq 0 \quad \alpha \geq 0 \quad \beta > 0 \\ E[Y] &= \alpha\beta \quad \text{Var}[Y] = \alpha\beta^2 \\ L((\alpha, \beta)|y) &= \left(\frac{1}{\Gamma(\alpha) \beta^\alpha} \right)^n \prod_{i=1}^n y_i^{\alpha-1} \exp\{-y_i/\beta\} \\ l((\alpha, \beta)|y) &= n \left(-\log(\Gamma(\alpha)) - \alpha \log(\beta) + (\alpha - 1) \overline{\log(y)} - \bar{y}/\beta \right) \end{aligned} \quad (1)$$

Função escore:

$$\begin{cases} \frac{dl}{d\beta} = n \left(-\frac{\alpha}{\beta} + \frac{\bar{y}}{\beta^2} \right) \\ \frac{dl}{d\alpha} = n \left(-\psi(\alpha) - \log(\beta) + \overline{\log y} \right) \end{cases} \quad (2)$$

em que $\psi(\alpha) = \frac{d}{d\alpha} \log(\Gamma(\alpha)) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ é a *função digama* (calculada no R com função `digamma()`).

Hessiano, Informação Observada e Esperada:

$$I_O(\alpha, \beta) = -H(\alpha, \beta) = - \begin{bmatrix} \frac{d^2 l}{d\beta^2} & \frac{d^2 l}{d\alpha d\beta} \\ \frac{d^2 l}{d\alpha d\beta} & \frac{d^2 l}{d\alpha^2} \end{bmatrix} = \begin{bmatrix} n \left(2 \frac{\bar{y}}{\beta^3} - \frac{\alpha}{\beta^2} \right) & \frac{n}{\beta} \\ \frac{n}{\beta} & n\psi'(\alpha) \end{bmatrix} \quad (3)$$

em que $\psi'(\alpha) = \frac{d^2}{dt^2} \log(\Gamma(\alpha))$ é a *função trigama* (calculada no R com função `trigamma()`).

$$I_E(\alpha, \beta) = E_Y[I_O(\alpha, \beta)] = \dots \quad (4)$$

Funções concentradas (perfilhadas):

Para este caso não existem expressões fechadas para o estimador de máxima verossimilhança. As funções dadas anteriormente são funções de dois parâmetros (α e β) mas, neste caso, é possível *reduzir a dimensionalidade* e escrevê-las como funções de apenas um parâmetro. Para isto considera-se o comportamento da função no estimador de máxima verossimilhança. Igualando a zero primeira função escore em 2 obtemos uma expressão fechada para um dos parâmetros dada por:

$$\hat{\alpha}\hat{\beta} = \bar{y}. \quad (5)$$

Pode-se então isolar um dos parâmetros e reescrever as funções de verossimilhança, escore e hessiana como funções de apenas um parâmetro. Substituindo β por $\hat{\beta} = \bar{y}/\hat{\alpha}$ temos que a log-verossimilhança (1) pode ser escrita na forma concentrada (perfilhada) como função apenas do parâmetro α :

$$pl(\alpha|y) = l_{\hat{\beta}}(\alpha|y) = n \left(-\log(\Gamma(\alpha)) - \alpha(\log(\bar{y}) - \log(\alpha)) + (\alpha - 1)\overline{\log(y)} - \alpha \right). \quad (6)$$

A função escore fica:

$$U_{\hat{\beta}}(\alpha) = n \left(-\psi(\alpha) - \log \bar{y} + \log(\alpha) + \overline{\log y} \right). \quad (7)$$

O hessiano para α é:

$$H_{\hat{\beta}}(\alpha) = n \left(-\psi'(\alpha) + \frac{1}{\alpha} \right). \quad (8)$$

E, neste caso, como a expressão não depende da v.a. Y as informações observadas e esperadas são iguais:

$$I_E(\alpha) = I_O(\alpha) = -H(\alpha)$$

Alternativamente, substituindo $\alpha = \hat{\alpha} = \bar{y}/\hat{\beta}$ pode-se obter expressões para o parâmetro β :

$$pl(\beta|y) = l_{\hat{\alpha}}(\beta|y) = \dots \quad (9)$$

$$U_{\hat{\beta}}(\alpha) = \dots H_{\hat{\beta}}(\alpha) = \dots I_O(\beta) = \dots I_E(\beta) = \dots \quad (10)$$

Obtenção das estimativas de máxima verossimilhança

O EMV é solução conjunta de

$$\begin{cases} \overline{\log y} - \log \beta + & = \psi(\bar{y}/\beta) \\ \hat{\alpha}\hat{\beta} & = \bar{y} \end{cases}$$

que neste caso devem ser obtidas por algoritmos numéricos. Diferentes algoritmos podem ser utilizados conforme será ilustrado mais adiante. Quatro opções serão consideradas:

1. algoritmos de maximização:

- (a) da log-verossimilhança de dois parâmetros,
 - (b) da log-verossimilhança concentrada (de um parâmetro) com posterior substituição em (5) para obtenção da estimativa do outro parâmetro
2. algoritmos de solução de equações (obtidas igualando a zero as funções escore);
- (a) do sistema de equações de dois parâmetros,
 - (b) do sistema concentrado (equação de um parâmetro) com posterior substituição em (5) para obtenção da estimativa do outro parâmetro.

Parametrização 2:

Rizzo (2008) e outros autores utilizam a parametrização com $r = \alpha$, $\lambda = 1/\beta$ em que λ é o parâmetro de taxa (*rate*). Esta é também a parametrização original do **S** e **S-Plus**.

Densidade, esperança, variância, (log)verossimilhança, escore e hessiano:

$$\begin{aligned}
Y &\sim G(r, \lambda) \\
f(y|r, \lambda) &= \frac{\lambda^r}{\Gamma(r)} y^{r-1} \exp\{-\lambda y\} \quad y \geq 0 \quad r \geq 0 \quad \lambda > 0 \\
E[Y] &= r/\lambda \quad Var[Y] = r/\lambda^2 \\
L((r, \lambda)|y) &= \left(\frac{\lambda^r}{\Gamma(r)} \right)^n \prod_{i=1}^n y_i^{r-1} \exp\{-\lambda y_i\} \\
l((r, \lambda)|y) &= n \left(r \log(\lambda) - \log(\Gamma(r)) + (r-1)\overline{\log(y)} - \lambda \bar{y} \right)
\end{aligned} \tag{11}$$

Função escore:

$$\begin{cases} \frac{dl}{d\lambda} = n \left(\frac{r}{\lambda} - \bar{y} \right) \\ \frac{dl}{dr} = n \left(\log(\lambda) - \frac{\Gamma'(r)}{\Gamma(r)} + \overline{\log y} \right) \end{cases}$$

Igualando as funções a zero, da primeira equação temos $\hat{\lambda} = \hat{r}/\bar{y}$. Substituindo λ por $\hat{\lambda}$ a segunda expressão é escrita como:

$$n \left(\log \frac{\hat{r}}{\bar{y}} + \overline{\log y} - \frac{\Gamma'(\hat{r})}{\Gamma(\hat{r})} \right) = 0$$

O *MLE* é solução conjunta de:

$$\begin{cases} \log \lambda + \overline{\log y} = \psi(\lambda \bar{y}) \\ \bar{y} = r/\lambda \end{cases} \tag{12}$$

em que $\psi(t) = \frac{d}{dt} \log(\Gamma(t)) = \frac{\Gamma'(t)}{\Gamma(t)}$ (função `digamma()`).

(Incluir Hessiano, informação e perfiladas como na parametrização anterior)

Parametrização 3:

Aitkin (2010) menciona duas parametrizações sendo a primeira, a mesma implementada no R, apontada como a mais usual, e uma segunda parametrizada por $r = \alpha$ e $\mu = \alpha\beta$.

Esta última parametrização tem propriedades interessantes para inferência. A primeira é a ortogonalidade entre r e μ na matriz de informação. Além disto em geral μ é o usualmente o parâmetro de interesse para inferências e r é um parâmetro *nuisance*. Finalmente a parametrização é adequada

para modelagem estatística na qual usualmente se propõe um modelo de regressão para média μ , como por exemplo em modelos lineares generalizados (*GLM*).

Densidade, esperança, variância e verossimilhança:

$$\begin{aligned}
Y &\sim G(r, \mu) \\
f(y|r, \mu) &= \frac{r^r}{\Gamma(r)\mu^r} y^{r-1} \exp\{-ry/\mu\} \quad y \geq 0 \quad r \geq 0 \quad \mu \geq 0 \\
E[Y] &= \mu \quad Var[Y] = \mu^2/r \\
L((r, \mu)|y) &= \left(\frac{r^r}{\Gamma(r)\mu^r} \right)^n \prod_{i=1}^n y_i^{r-1} \exp\{-ry_i/\mu\} \\
l((r, \mu)|y) &= n \left(r(\log(r) - \log(\mu)) - \log(\Gamma(r)) + (r-1)\overline{\log(y)} - \frac{r}{\mu}\bar{y} \right)
\end{aligned}$$

Função escore:

$$\begin{cases} \frac{dl}{d\mu} = n \left(-\frac{r}{\mu} + \frac{r\bar{y}}{\mu^2} \right) \\ \frac{dl}{dr} = n \left(\log(r) + 1 - \log(\mu) - \frac{\Gamma'(r)}{\Gamma(r)} + \overline{\log y} - \frac{\bar{y}}{\mu} \right) \end{cases}$$

Igualando as funções a zero, da primeira equação temos $\hat{\mu} = \bar{y}$. Substituindo μ por $\hat{\mu}$ a segunda expressão é escrita como:

$$n \left(\log(\hat{r}) + 1 - \log(\bar{y}) - \frac{\Gamma'(\hat{r})}{\Gamma(\hat{r})} + \overline{\log y} - 1 \right) = 0$$

O *MLE* é solução conjunta de:

$$\begin{cases} \log \hat{r} - \psi(\hat{r}) = \log \bar{y} - \overline{\log y} \\ \hat{\mu} = \bar{y} \end{cases}$$

em que $\psi(t) = \frac{d}{dt} \log(\Gamma(t)) = \frac{\Gamma'(t)}{\Gamma(t)}$ (calculado no R pela função `digamma()`).

ToDo:(Incluir Hessiano, informação e perfiladas como na parametrização anterior)

1.2 Códigos R para obtenção das estimativas

Não é possível obter *MLE* em forma analítica fechada para ambos parâmetros da distribuição Gama sendo então necessário o uso de métodos numéricos para obter estimativas. Algumas abordagens numéricas possíveis são:

1. *via* solução de equações (função escore)
 - a. conjunta para dois parâmetros: resolução numérica do sistema de duas equações
 - b. por substituição: resolução numérica de uma equação para encontrar a estimativa de um parâmetro que é substituída na equação conhecida para obter a estimativa do outro parâmetro.
2. *via* maximização de função de log-verossimilhança
 - a. maximização bidimensional (encontra numericamente estimativas dos 2 parâmetros)

- b. unidimensional (verossimilhança concentrada/perfilhada para 1 parâmetro, seguida de substituição para o outro parâmetro)

Gerando dados (simulados).

```
set.seed(201107)
dadosG <- rgamma(20, shape = 4.5, rate=2)
```

Estatísticas amostrais que são as quantidades calculadas com os dados utilizadas nas avaliações das funções.

```
am <- list(media=mean(dadosG),
           media.logs = mean(log(dadosG)),
           n=length(dadosG))
```

Estimação utilizando métodos numéricos utilizando a **Parametrização 2**.

(1.a.) Equações de estimação Definimos uma função em R com sistema de equações definido em 12. As estimativas são dadas pela solução deste sistema que é obtida utilizando a função `multiroot()` do pacote **rootSolve**.

```
Umat <- function(par, amostra){
  lambda <- par[1] ; r = par[2]
  UlamUr <- local({
    c(n * ((r/lambda) - media),
      n*(log(lambda) - digamma(lambda*media) + media.logs))
  }, env=amostra)
  return(UlamUr)
}
rootSolve::multiroot(f=Umat, start=c(1,1), amostra=am)[["root"]]
## [1] 2.408413 4.692590
```

(1.b.) Equações de estimação - substituição

$$\left(\log \frac{\hat{r}}{\bar{y}} + \overline{\log y} - \frac{\Gamma'(r)}{\Gamma(r)} \right) = 0 \quad (\text{numericamente})$$
$$\hat{\lambda} = \hat{r}/\bar{y} \quad (\text{substituição})$$

A função `uniroot()` pode ser usada para resolver numericamente uma equação.

```
Ulam <- function(lambda, amostra)
  with(amostra, digamma(lambda*media) - media.logs - log(lambda))
(lambda.est <- uniroot(Ulam, lower=1e-3, upper=1e5, amostra=am)$root)
## [1] 2.40842
(r.est <- lambda.est * am$media)
## [1] 4.692604
```

```
hist(dadosG, prob=T, xlim=c(0,5), main="")
lines(density(dadosG));rug(dadosG)
curve(dgamma(x, shape = 4.5, rate=2), 0, 5, add=T, lwd=2)
```

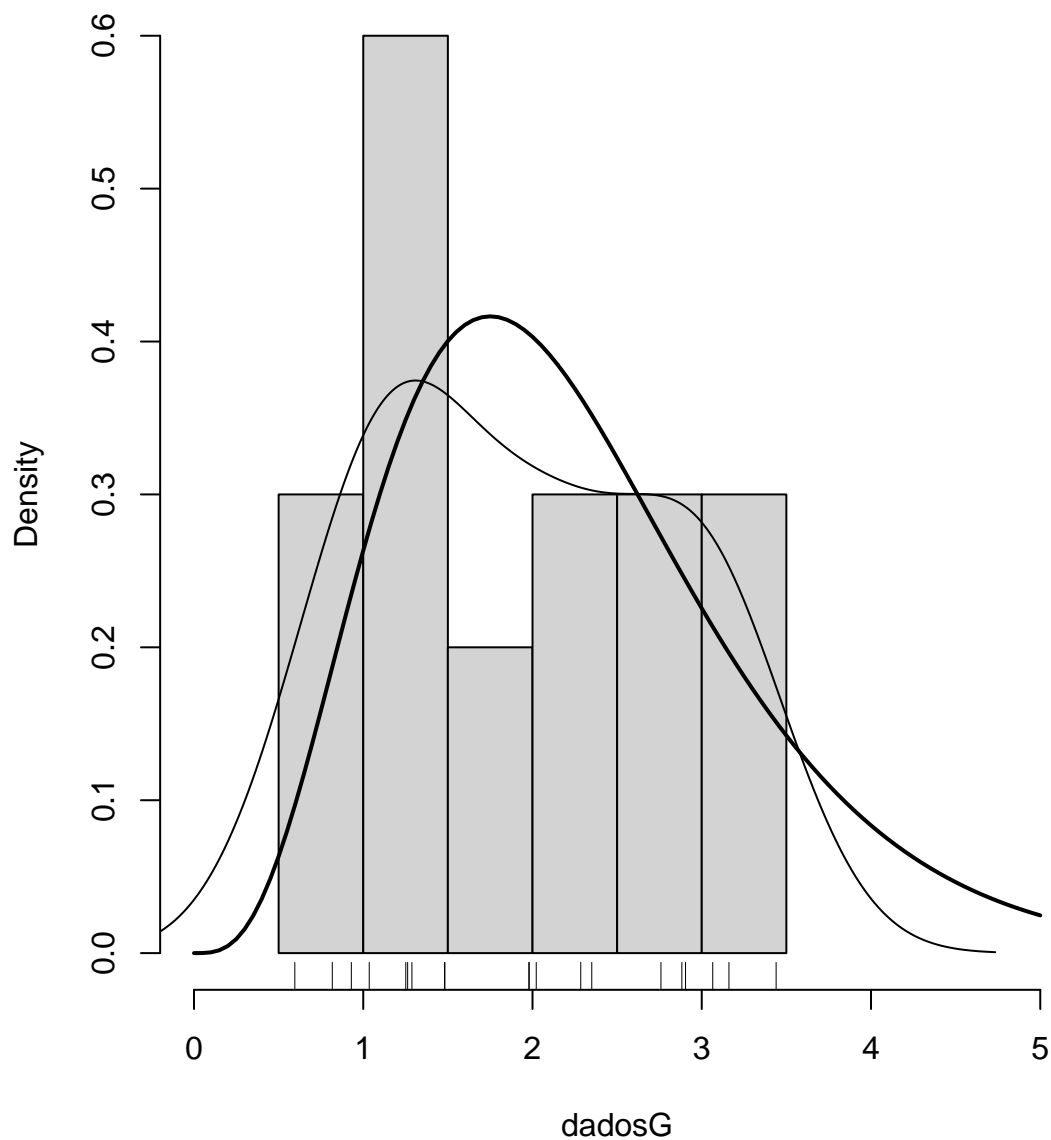


Figura 1: Histograma, densidade empírica e *rug* dos dados originais e densidade da distribuição simulada (linha grossa).

(2.a.) Maximização da função de log-verossimilhança (2 parâmetros) A função de (negativo) log-verossimilhança pode ser definida no R como uma função a 2 parâmetros.

```

neglLik <- function(par, amostra, modelo=2){
  if(modelo == 1){
    alpha <- par[1]; beta <- par[2]
    ll <- with(amostra, n*(-alpha*log(beta) - log(gamma(alpha)) +
      (alpha-1) * media.logs - media/beta))
  }
  if(modelo == 2){
    r <- par[1]; lambda <- par[2]
    ll <- with(amostra, n*(r*log(lambda) - log(gamma(r)) +
      (r-1) * media.logs - lambda * media))
  }
  if(modelo == 3){
    r <- par[1]; mu <- par[2]
    ll <- with(amostra, n*(r*(log(r) - log(mu)) - log(gamma(r)) +
      (r-1) * media.logs - (r/mu) * media))
  }
  return(-ll)
}

```

A seguir é mostrada a obtenção das estimativas em cada parametrização. Para impor limites no espaço paramétrico utiliza-se os argumentos `lower` e `upper` e `method="L-BFGS-B"`.

```

(mod1 <- optim(c(1,1), neglLik, amostra=am, method="L-BFGS-B", modelo=1,
  lower=c(0,0), upper=c(Inf, Inf)))[["par"]]

## [1] 4.6924351 0.4152265

(mod2 <- optim(c(1,1), neglLik, amostra=am, method="L-BFGS-B", modelo=2,
  lower=c(0,0), upper=c(Inf, Inf)))[["par"]]

## [1] 4.692593 2.408415

(mod3 <- optim(c(1,1), neglLik, amostra=am, method="L-BFGS-B", modelo=3,
  lower=c(0,0), upper=c(Inf, Inf)))[["par"]]

## [1] 4.692591 1.948416

```

Obtenção de estimativas em uma parametrização a partir das estimativas de outra parametrização é obtida diretamente.

```

1/mod1[2]

## [1] 2.408324

prod(mod1)

## [1] 1.948423

```

Comentários sobre reparametrização

- Reparametrização para garantir validade do espaço paramétrico

No exemplo da Gama, ambos parâmetros devem ser não negativos em todas as parametrizações. Alternativamente a usar `lower` e `upper`, pode-se redefinir a função com a opção de que os parâmetros sejam fornecidos em escala logarítmica e desta forma possam assumir valores nos reais. Note que isto exclui o valor nulo do espaço paramétrico original. Aproveitamos e redefinição da função de (negativo) log-verossimilhança para ilustrar syntaxes em R escrevendo a função usando `switch()` para definir a opção de parametrização.

```
neglLik <- function(par, amostra, modelo=2, logpar=FALSE){
  if(logpar) par <- exp(par)
  ll <- switch(modelo,
    "1" = {alpha <- par[1]; beta <- par[2];
           with(amostra, n*(-alpha*log(beta)-log(gamma(alpha)) +
                    (alpha-1) * media.logs - media/beta))},
    "2" = {alpha <- par[1]; lambda <- par[2] ;
           with(amostra, n*(alpha*log(lambda)-log(gamma(alpha)) +
                    (alpha-1) * media.logs - lambda * media))},
    "3" = {alpha <- par[1]; mu <- par[2] ;
           with(amostra, n*(alpha*(log(alpha)-log(mu))-log(gamma(alpha)) +
                    (alpha-1) * media.logs - (alpha/mu) * media))})
  return(-ll)
}
```

Obtendo novamente as estimativas (nas escalas logarítmicas e original) agora com o método padrão da função `optim()`.

```
(mod1r <- optim(log(c(1,1)), neglLik, amostra=am, modelo=1, log=T)$par)

## [1] 1.5460915 -0.8791401

rbind(mod1, exp(mod1r))

##           [,1]      [,2]
## mod1 4.692435 0.4152265
##      4.693091 0.4151398

(mod2r <- optim(log(c(1,1)), neglLik, amostra=am, modelo=2, log=T)$par)

## [1] 1.5458999 0.8788993

rbind(mod2, exp(mod2r))

##           [,1]      [,2]
## mod2 4.692593 2.408415
##      4.692192 2.408247
```



```
(mod3r <- optim(log(c(1,1)), neglLik, amostra=am, modelo=3, log=T)$par)

## [1] 1.5458160 0.6670942

rbind(mod3, exp(mod3r))

##           [,1]      [,2]
## mod3 4.692591 1.948416
##      4.691798 1.948567
```

- Reparametrização para interpretabilidade dos parâmetros
COMENTAR
- Reparametrização ortogonalização
COMENTAR
- Melhor comportamento (quadrático) da função.
COMENTAR. Invariância dos estimadores de M.V. e transformação direta de limites de intervalos de confiança baseados na verossimilhança ou *deviance*.

(2.b.) Maximização da função de log-verossimilhança concentrada (1 parâmetro)

Quando possível é conveniente (numericamente) reduzir a dimensionalidade da maximização numérica utilizando a verossimilhança concentrada/perfilhada. No exemplo substituímos a expressão do estimador que possui forma fechada $\hat{\lambda} = \hat{r}/\bar{y}$ na função verossimilhança (11) e obtemos a função de verossimilhança em função de apenas um parâmetro.

$$l(r|y) = n \left(r \log(\hat{\lambda}) - \log(\Gamma(r)) + (r-1) \overline{\log(y)} - \hat{\lambda} \bar{y} \right) \quad (13)$$

$$= n \left(r \log(r/\bar{y}) - \log(\Gamma(r)) + (r-1) \overline{\log(y)} - r \right) \quad (14)$$

Esta função é definida e usada para obter a estimativa de r que na sequência é substituída em $\hat{\lambda} = \hat{r}/\bar{y}$ para obter a estimativa de λ .

```
neglLik1 <- function(par, amostra, modelo=2, logpar=FALSE){
  if(logpar) par <- exp(par)
  ll <- switch(modelo,
    "1" = {alpha <- amostra$media/par; beta <- par;
      with(amostra, n*(-log(gamma(alpha)) - alpha*log(beta) +
        (alpha-1) * media.logs - media/beta))},
    "2" = {alpha <- par * amostra$media; lambda <- par;
      with(amostra, n*(alpha*log(lambda) - log(gamma(alpha)) +
        (alpha-1) * media.logs - lambda * media))},
    "3" = {with(amostra, n*(par*(log(par)-log(media)) -
      log(gamma(par)) + (par-1)*media.logs-par))})
  return(-ll)
}
```

Como a otimização agora é em apenas uma dimensão utiliza-se a função `optimize()`. As estimativas em cada parametrização são obtidas com:

```
(est1 <- optimize(neglLik1, lower=0.001, upper=100, amostra=am, modelo=1)$min)
## [1] 0.4152092

(est2 <- optimize(neglLik1, lower=0.001, upper=100, amostra=am, modelo=2)$min)
## [1] 2.408401

(est3 <- optimize(neglLik1, lower=0.001, upper=100, amostra=am, modelo=3)$min)
## [1] 4.692589
```

mas, na prática, basta obter em uma dessas e as estimativas em outras parametrizações são obtidas de forma trivial.

```
1/est1
## [1] 2.408424

am$media/est1
## [1] 4.692613

am$media*est2
## [1] 4.692568
```

1.3 Uso de algumas funções do R

O R e seus pacotes oferece funções que facilitam os procedimentos acima, bem como cálculos posteriores como intervalos, matriz de covariâncias, dentre outras. Dentre as funções "facilitadoras" já prontas e disponíveis estão a `mle()` do pacote **stats4** e `mle2()` do pacote **bbmle**. Nestas funções os parâmetros são passados na forma de listas.

Inicialmente define-se a função de (negativo)log-verossimilhança

```
nlLik <- function(r, lambda){
  lambda <- exp(lambda); r <- exp(r)
  ll <- am$n*(r*log(lambda) - log(gamma(r)) +
    (r-1)*am$media.logs - lambda*am$media)
  return(-ll)
}
```

que é maximizada para obter as estimativas.

```

require(stats4)
args(mle)

## function (minuslogl, start, optim = stats::optim, method = if (!useLim) "BFGS" else "L-BFGS-B",
##      fixed = list(), nobs, lower, upper, ...)
## NULL

(est.mle1 <- mle(nlLik, start=list(r=log(2), lambda=log(2)),
                method="L-BFGS-B", lower=c(0,0)))

##
## Call:
## mle(minuslogl = nlLik, start = list(r = log(2), lambda = log(2)),
##      method = "L-BFGS-B", lower = c(0, 0))
##
## Coefficients:
##      r      lambda
## 1.545982 0.878965

```

Vários resumos e funções extratoras estão imediatamente disponíveis para saídas da `mle()`.

```

## resumo
summary(est.mle1)

## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = nlLik, start = list(r = log(2), lambda = log(2)),
##      method = "L-BFGS-B", lower = c(0, 0))
##
## Coefficients:
##      Estimate Std. Error
## r      1.545982  0.3056496
## lambda 0.878965  0.3226093
##
## -2 log L: 49.52223

## valor da verossimilhança maximizada
logLik(est.mle1)

## 'log Lik.' -24.76112 (df=2)

## AIC
AIC(est.mle1)

## [1] 53.52223

## estimativas
coef(est.mle1)

```

```
par(mfrow=c(1,2))
plot(profile(est.mle1))
```

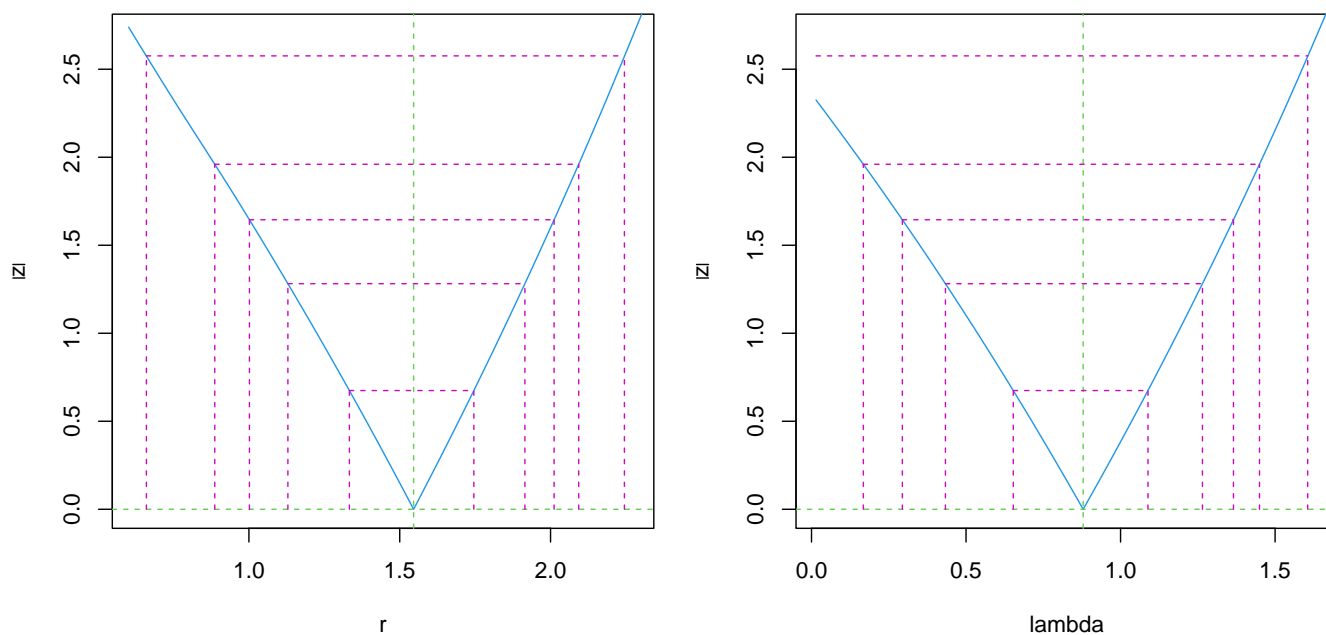


Figura 2: Perfis de verossimilhança (na escala de raiz da deviance) para parâmetros do modelo Gama sob a parametrização 2.

```
##      r      lambda
## 1.545982 0.878965

## matriz de covariâncias
vcov(est.mle1)

##      r      lambda
## r      0.09342171 0.09342167
## lambda 0.09342167 0.10407675

## IC baseado na verossimilhança (deviance) perfilhada
confint(est.mle1, level=0.95)

## Profiling...
##      2.5 %    97.5 %
## r      0.8868132 2.093177
## lambda 0.1676132 1.449356
```

Assim como funções gráficas também são definidas como por exemplo os perfis de verossimilhança (expressos como raízes da função deviance) mostrados na Figura 2

1.4 Visualizações da função de verossimilhança

A seguir vamos visualizar as superfícies de verossimilhança para cada uma das três parametrizações.

Uma função mais conveniente para visualização é a função **deviance** definida por:

$$D(\theta) = -2[l(\theta) - l(\hat{\theta})]$$

que é definida como uma função no R:

```
devFun <- function(theta, est, llFUN, ...){  
  return(2 * (llFUN(theta, ...) - llFUN(est, ...)))  
}
```

A seguir tomamos as estimativas e verificamos que as funções estão retornando valores corretamente.

```
(ests <- c(r.est, lambda.est))  
  
## [1] 4.692604 2.408420  
  
## verossimilhanças originais (2 parâmetros)  
neglLik(ests, amostra=am, modelo=2)  
  
## [1] 24.76112  
  
neglLik(log(ests), amostra=am, modelo=2, logpar=T)  
  
## [1] 24.76112  
  
## verossimilhanças concentradas  
neglLik1(ests[2], amostra=am, modelo=2)  
  
## [1] 24.76112  
  
neglLik1(log(ests[2]), amostra=am, modelo=2, logpar=T)  
  
## [1] 24.76112  
  
## fc deviance (2 parâmetros)  
devFun(ests, est=mod2, llFUN=neglLik, amostra=am, modelo=2)  
  
## [1] 8.101608e-11  
  
devFun(log(ests), est=mod2r, llFUN=neglLik, amostra=am, modelo=2, logp=T)  
  
## [1] -1.014605e-07  
  
## fc deviance concentrada (1 parâmetro)  
devFun(ests[2], est=mod2[2], llFUN=neglLik1, amostra=am, modelo=2)  
  
## [1] 8.057555e-11
```

```

devFun(log(ests[2]), est=mod2r[2], llFUN=negLik1, amostra=am, modelo=2, logp=T)

## [1] -5.047529e-08

## agora para outros valores que não soas os EMV
negLik(c(4,2.5), amostra=am, modelo=2)

## [1] 26.55084

negLik(log(c(4,2.5)), amostra=am, modelo=2, logpar=T)

## [1] 26.55084

negLik1(2.5, amostra=am, modelo=2)

## [1] 24.76866

negLik1(log(2.5), amostra=am, modelo=2, logpar=T)

## [1] 24.76866

devFun(c(4,2.5), est=mod2, llFUN=negLik, amostra=am, modelo=2)

## [1] 3.579439

devFun(log(c(4,2.5)), est=mod2r, llFUN=negLik, amostra=am, modelo=2, logp=T)

## [1] 3.579439

devFun(2.5, est=mod2[2], llFUN=negLik1, amostra=am, modelo=2)

## [1] 0.01508588

devFun(log(2.5), est=mod2r[2], llFUN=negLik1, amostra=am, modelo=2, logp=T)

## [1] 0.01508583

```

Para obter os gráficos da função de verossimilhança uni e bidimensional é conveniente vetorizar a função para que possa receber uma sequencia de valores do(s) parametro(s) que serão usados na construção do gráfico (isto é equivalente a fazer um *for-loop*). A função deviance vetorizada é definida como se segue e conferimos com os valores obtidos anteriormente.

```

Ofun <- Vectorize(function(x,y, ...) devFun(c(x,y), ...))
Ofun(4, 2.5, est=ests, llFUN=negLik, amostra=am, modelo=2)

## [1] 3.579439

Ofun(log(4), log(2.5), est=log(ests), llFUN=negLik, amostra=am, modelo=2, logpar=T)

## [1] 3.579439

```

Para obter o gráfico da função deviance a dois parâmetros, definimos uma malha de valores e avaliamos a função a cada ponto da malha. Note que vamos fazer na escala original e também logarítmica dos parâmetros. Começamos com a **parametrização 1** e as superfícies são mostradas na Figura 3.

```
alphaGR <- seq(1.6,10.5, len=100)
betaGR <- seq(0.17,1.35, len=100)
GRdev <- outer(alphaGR, betaGR, FUN = Ofun, est=mod1, llFUN=negllik, amostra=am, modelo=1)

alphaGRl <- seq(log(1.6),log(10.5), len=100)
betaGRl <- seq(log(0.17),log(1.35), len=100)
GRdevl <- outer(alphaGRl, betaGRl, FUN = Ofun, est=log(mod1), llFUN=negllik, amostra=am, mo
```

Passamos agora à **parametrização 2** e após a definição da malha e avaliação da função, as superfícies são mostradas na Figura 4.

```
rGR <- seq(1.6,10.7, len=100)
lambdaGR <- seq(0.7,5.6, len=100)
GRdev <- outer(rGR, lambdaGR, FUN=Ofun, est=mod2, llFUN=negllik, amostra=am)

rGRl <- seq(log(1.6),log(10.7), len=100)
lambdaGRl <- seq(log(0.7),log(5.6), len=100)
GRdevl <- outer(rGRl, lambdaGRl, FUN=Ofun, est=log(mod2), llFUN=negllik, amostra=am, logpar
```

O mesmo procedimento é repetido para **parametrização 3** com superfícies mostradas na Figura 5.

```
rGR <- seq(1.5,11, len=100)
muGR <- seq(1.4,2.83, len=100)
GRdev <- outer(rGR, muGR, FUN = Ofun, est=mod3, llFUN=negllik, amostra=am, modelo=3)

rGRl <- seq(log(1.5),log(11), len=100)
muGRl <- seq(log(1.4),log(2.83), len=100)
GRdevl <- outer(rGRl, muGRl, FUN = Ofun, est=log(mod3), llFUN=negllik, amostra=am, modelo=3)
```

Gráficos feitos utilizando a reparametrização em escala logarítmica reduzem a assimetria na função. A última parametrização apresenta uma superfície mais próxima da ortogonalidade.

Unidimensional, verossimilhança perfilhada

```
#devFun(lambdaGR, est=ests[2], llFUN=negllik1, amostra=am)
curve(devFun(x, est=ests[2], llFUN=negllik1, amostra=am), from=0.9, to=5,
      xlab=expression(lambda), ylab=expression(Pl(lambda)))
## cortes definidos por probabilidades
(corteICs <- qchisq(c(0.90, 0.95, 0.99), df=1))

## [1] 2.705543 3.841459 6.634897

text(4.5, corteICs, c("90%", "95%", "99%"), pos=3)

## cortes definidos por percentuais da MV
(corteICs <- -2*log(c(0.25, 0.15, 0.04)))
```

```

par(mfrow=c(2,2))
LEVELS <- c(0.99,0.95,0.9,0.7,0.5,0.3,0.1,0.05)
contour(alphaGR,betaGR, GRdev, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(alpha),ylab=expression(beta))
points(t(mod1), pch=19, cex=0.6)
persp(alphaGR, betaGR, GRdev)

contour(alphaGRl,betaGRl, GRdevl, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(log(alpha)),ylab=expression(log(beta)))
points(t(log(mod1)), pch=19, cex=0.6)
persp(alphaGRl, betaGRl, GRdevl)

```

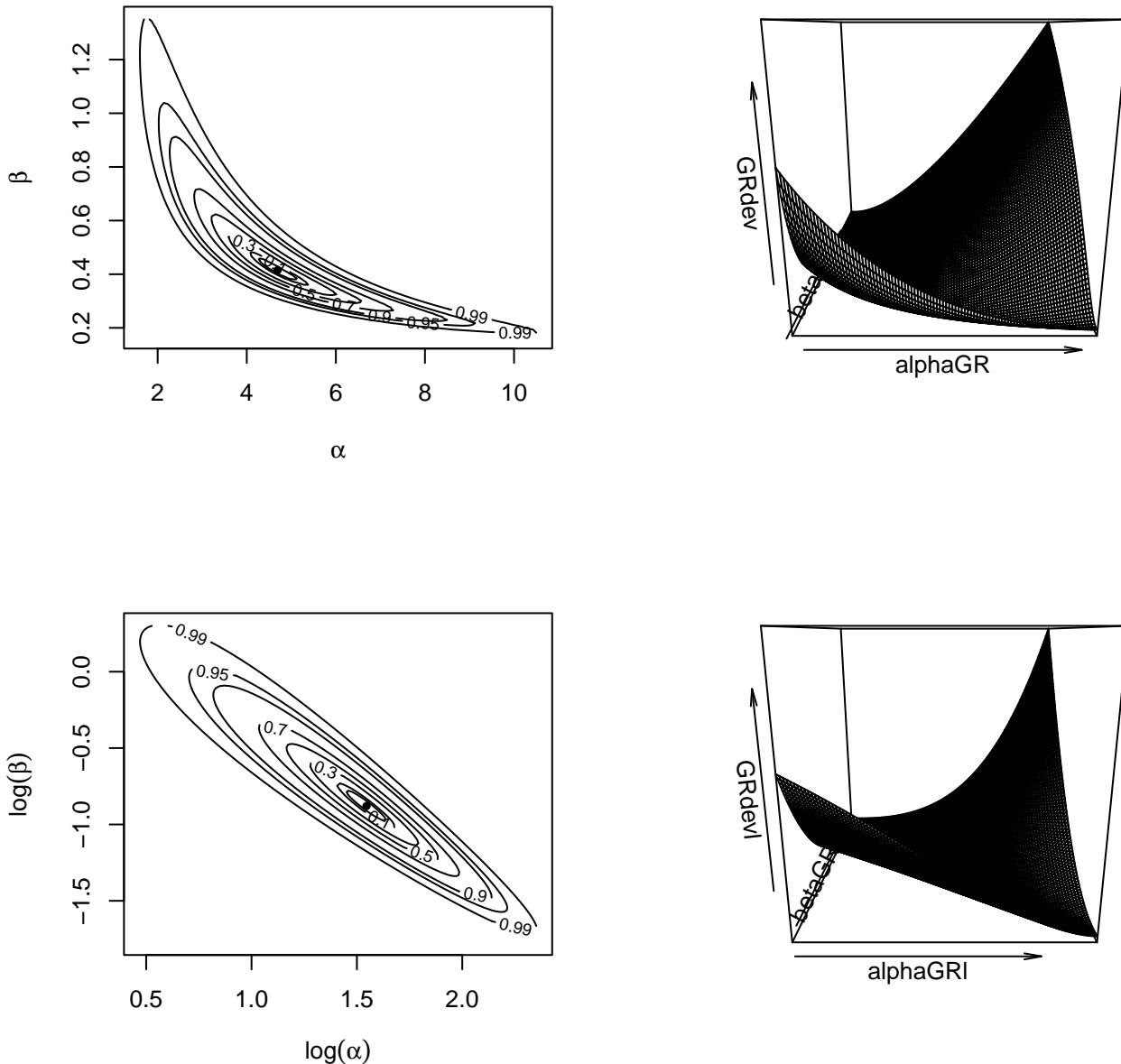


Figura 3: **Parametrização 1:** superfícies de verossimilhanças (expressa como deviance) a dois parâmetros. Gráficos superiores com parâmetros originais e inferiores na escala logarítmica.


```

par(mfrow=c(2,2))
contour(rGR,lambdaGR, GRdev, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(r),ylab=expression(lambda))
points(t(mod2), pch=19, cex=0.6)
persp(rGR, lambdaGR, GRdev)

contour(rGRl,lambdaGRl, GRdevl, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(log(r)),ylab=expression(log(lambda)))
points(t(log(mod2)), pch=19, cex=0.6)
persp(rGRl, lambdaGRl, GRdevl)

```

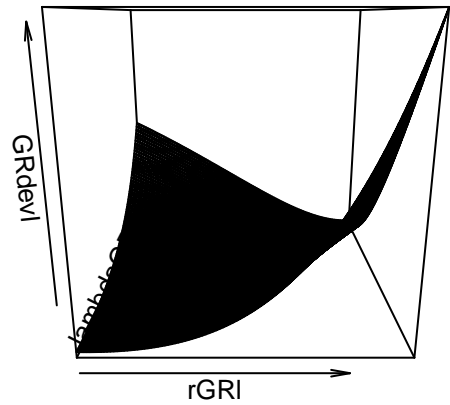
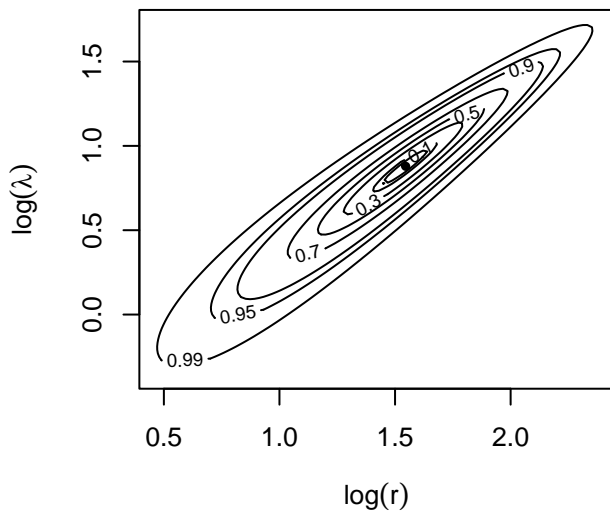
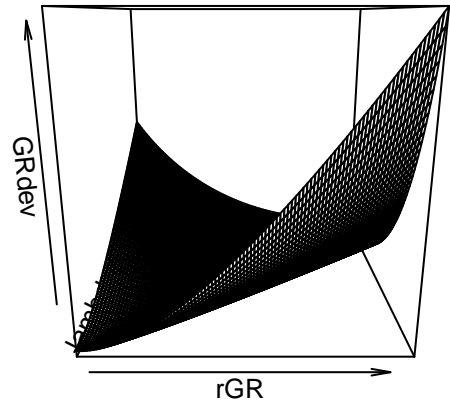
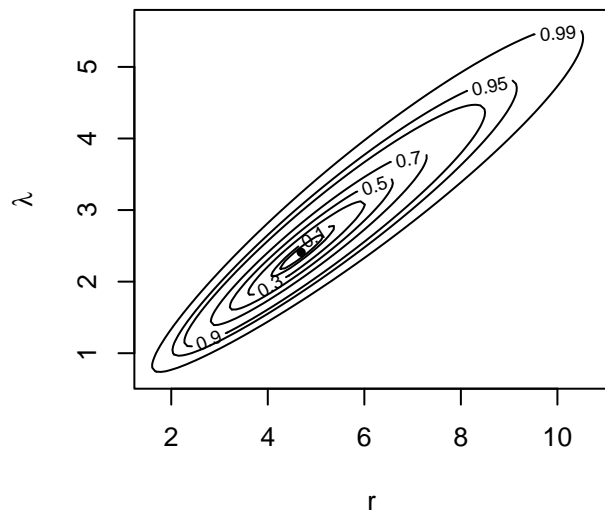


Figura 4: **Parametrização 2**: superfícies de verossimilhanças (expressa como deviance) a dois parâmetros. Gráficos superiores com parâmetros originais e inferiores na escala logarítmica.

```

par(mfrow=c(2,2))
contour(rGR,muGR, GRdev, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(r),ylab=expression(lambda))
points(t(mod3), pch=19, cex=0.6)
persp(rGR, muGR, GRdev)

contour(rGRl, muGRl, GRdevl, levels=qchisq(LEVELS,df=2),
        labels=LEVELS, xlab=expression(log(r)),ylab=expression(log(lambda)))
points(t(log(mod3)), pch=19, cex=0.6)
persp(rGRl, muGRl, GRdevl)

```

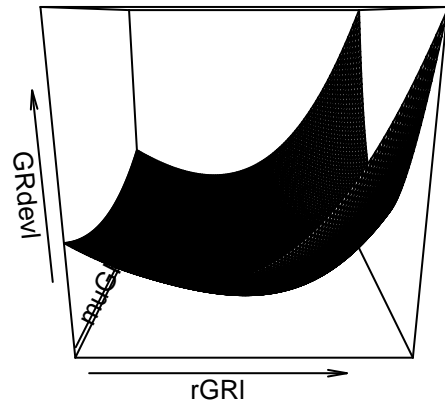
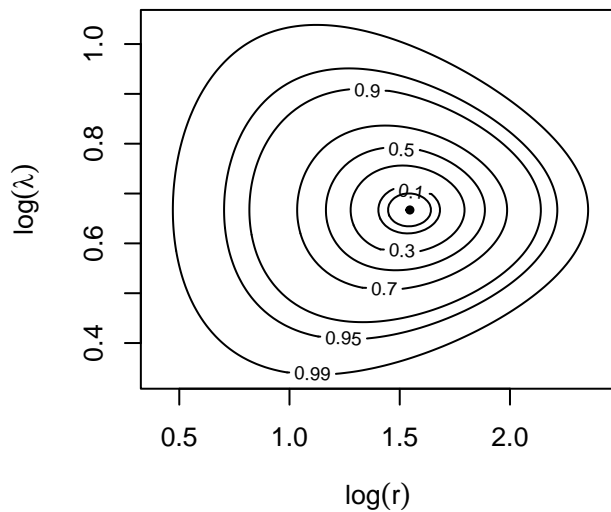
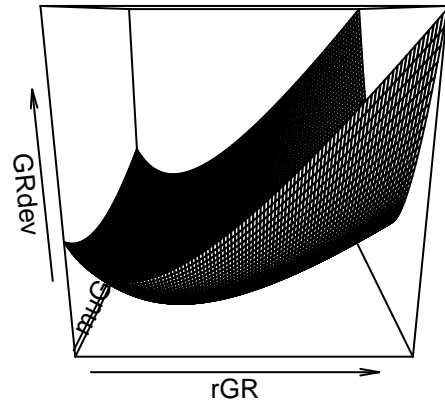
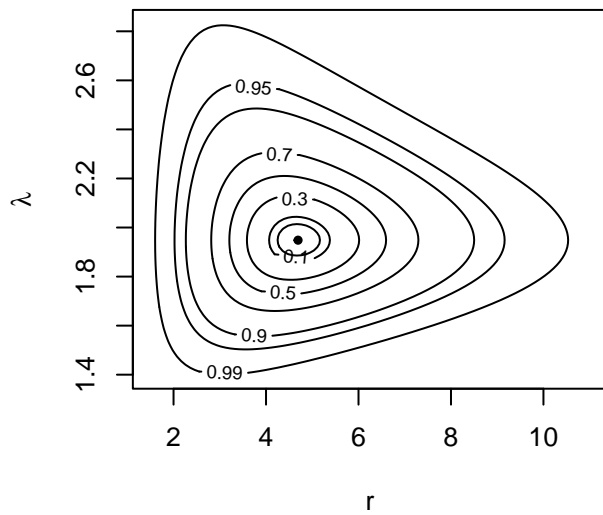


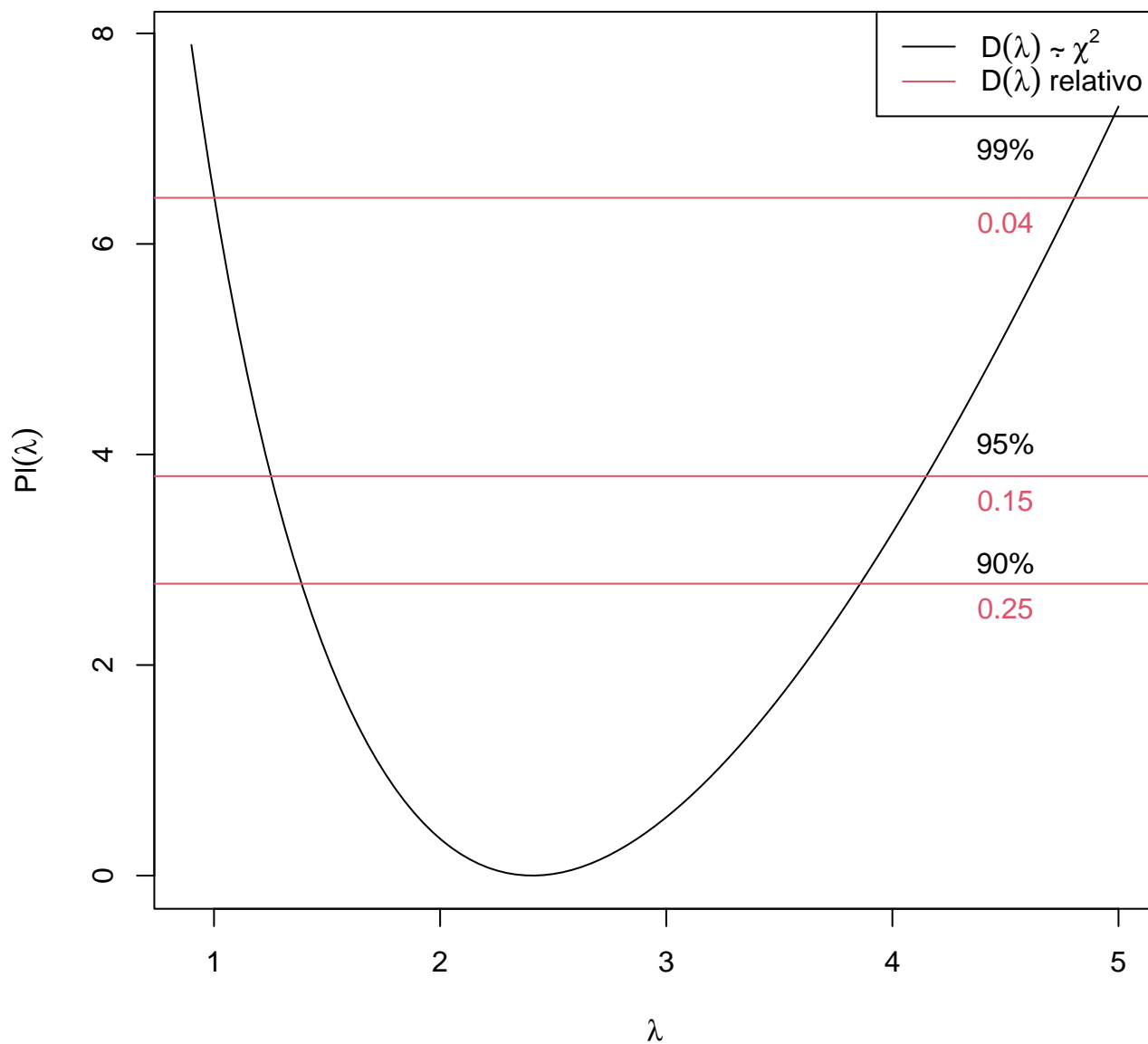
Figura 5: **Parametrização 3**: superfícies de verossimilhanças (expressa como deviance) a dois parâmetros. Gráficos superiores com parâmetros originais e inferiores na escala logarítmica.

```
## [1] 2.772589 3.794240 6.437752
```

```
abline(h=corteICs, col=2)
```

```
text(4.5, corteICs, c(0.25, 0.15, 0.04), pos=1, col=2)
```

```
legend("topright", c(expression(D(lambda)~tilde(.)~chi^2),  
                      expression(D(lambda)~relativo)), lty=1, col=1:2)
```



Os gráficos e funções poderiam ser obtidos em escala logarítmica para os parâmetros e os intervalos por transformação dos limites dos IC's baseados em verossimilhança, dada a propriedade de invariância.

Aproximação Quadrática - Aproximação de Taylor de 2a ordem da função verossimilhança e *deviance* ao redor do estimador de M.V. $\hat{\theta}$.

$$\begin{aligned}l(\theta) &= l(\hat{\theta}) + (\theta - \hat{\theta})l'(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^2l''(\hat{\theta}) \\ &= l(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^2l''(\hat{\theta})\end{aligned}$$

Uma vez que pela definição de E.M.V. $l'(\hat{\theta}) = 0$. A aproximação da função *deviance* fica:

$$\begin{aligned}D(\theta) &= -2[l(\theta) - l(\hat{\theta})] \\ &\approx -2[l(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^2l''(\hat{\theta}) - l(\hat{\theta})] \\ &= -(\theta - \hat{\theta})^2l''(\hat{\theta})\end{aligned}$$

As expressões consideram θ escalar, mas a extensão para um vetor de parâmetros é trivial.

Complementos: falta ainda completar texto com alguns tópicos como:

- Mostrar como fica a aproximação quadrática em cada parametrização em 1-D (perfilhada) e 2-D
- obter Io/Ie para cada parametrização
- Comparar Io/Ie com o hessiano numérico
- Escrever o algoritmo Newton-Rapson
- escrever e usar as funções gradiente na otimização para funções 1-D e 2-D
- ilustrar obtenção dos intervalos por diferentes métodos (incluindo simulação para mostrar taxas de cobertura)
- ...